Using OWA Operators for Gene Sequential Pattern Clustering

Jordi Nin[1*] [1]LAAS-CNRS jnin@laas.fr
Paola Salle[2] [2]LIRMM salle@lirmm.fr
Sandra Bringay[2,3] [3] U. Montpellier 3 bringay@lirmm.fr
Maguelonne Teisseire[2,4] [4]UMR TETIS, Cemagref teisseire@lirmm.fr

Abstract

Nowadays, the management of sequential patterns data becomes an increasing need in biological knowledge discovery processes. An important task in these processes is the restitution of the results obtained by using data mining methods. In a complex domain as biomedical, an efficient interpretation of the patterns without any assistance is difficult. One of the most common knowledge discovery proces is clustering. But the application of clustering to gene sequential patterns is far from easy on biomedical data.In this paper, we introduce a new gene sequential patterns similarity function and summarization algorithm.

1 Introduction

The DNA microarrays are a technology for gene expression analysis [3]. It is used to answer a large variety of biomedical questions such as gene expression profiling, comparative genomic hybridization and many others.

In order to analyze data produced by the DNA microarrays, there are several data mining methods. One of the most popular is the extraction of sequential patterns [1]. They outline relationships between genes by focusing on ordered expressions. For instance, the sequential pattern $S_{75}\% =< (G_A)(G_B)(G_C) >$ means that for 75% of the studied DNA microarrays, the gene G_A has an intensity smaller than the genes G_B and G_C.

As most of data mining methods, an efficient interpretation of the results is a difficult and challenging task.

*The reseach done in this work is funded by the European Community through the 7th Framework Programme Marie Curie Intra-European fellowship, contract No 235226. Partial support by the Spanish MEC (projects ARES – CONSOLIDER INGENIO 2010 CSD2007-00004 – and eAEGIS – TSI2007-65406-C03-02) is also acknowledged.

Moreover, in many cases, some patterns are very similar and they can be grouped together without loosing generality. For example, the patterns $M_1 =< (G_A)(G_B)(G_C) >$ and $M_2 =< (G_D)(G_B)(G_C) >$ differ only by the first gene (G_A is replaced by G_D) and it is also the least intense gene. For this reason, a keypoint for supporting the discovery process is to help experts organizing the large amount of sequential patterns into a smaller number of groups of meaningful patterns.

In this paper, We propose (1) a new similarity measure, called OWAGen, based on OWA operators for gene sequential pattern clustering; (2) a summarization process to associate a more general sequential pattern to each group (or cluster) of gene sequential patterns.

2 OWA Operator Definition

Aggregation functions are functions used for information fusion. Here, we consider the Ordered Weighted Aggregation (OWA) operator [5].

Definition 1 *A function* $Q : [0,1] \rightarrow [0,1]$ *is a* regular monotonically non-decreasing fuzzy quantifier *(non-decreasing fuzzy quantifiers) if it satisfies: (i)* $Q(0) = 0$*; (ii)* $Q(1) = 1$*; (iii)* $x > y$ *implies* $Q(x) \geq Q(y)$.

An example of family of fuzzy quantifiers is $Q^\alpha(x) = x^\alpha$ for $\alpha > 0$. Such family corresponds to Yager α-quantifiers. Using fuzzy quantifiers, the OWA operator [5] is defined as follows.

Definition 2 *Let* Q *be a non-decreasing fuzzy quantifier, then* $OWA_Q : \mathbb{R}^N \rightarrow \mathbb{R}$ *is an* (OWA) operator *if*

$$OWA_Q(a_1, ..., a_N) = \sum_{i=1}^{N} (Q(i/N) - Q((i-1)/N)) a_{\sigma(i)}$$

978-1-4244-4878-4/09/$25.00 ©2009 IEEE

where σ is a permutation such that $a_{\sigma(i)} \geq a_{\sigma(i+1)}$.

The interest of the OWA operators is that they permit the user to aggregate the values giving importance to large (or small) values.

3 Basic Ingredients

Here, we describe our new algorithm for gene sequential patterns summarization and our new similarity function.

3.1 Gene Sequential Patterns Summary

Aggregation (or summarization) functions combine N data values into a single one. There are a lot of research considering this problem for numerical or ordinal values. But, there is no work about considering the summarization of a set of gene sequential patterns. In this section, we introduce a new procedure to combine several gene sequential patterns into a more general one.

Firstly, one has to decide which information has to be preserved in the summary. In our case, we want to preserve the following one: (i)*Common genes in the same position*. If two gene sequential patterns have the same genes in the same position, it is telling to us that the intensity of these genes is usually ordered in the same way in several experiments. Such order can be typical of a particular biological condition. (ii)*Common genes in different positions*. This fact is giving us another important information, the intensity of these genes is usually ordered in different ways. However, there is a relationship between these common genes and certain biological condition. (iii) *Non common genes*. Clearly, non common genes are not so important as common ones. However, the existence of these non common genes has to be denoted in the summary. For this reason, we replace all non common genes by a wild-card. These wild-cards indicate to us that in the aggregated sequences there are several genes that can appear in the same position.

Our proposal is described in the Algorithm 1. It starts with an empty summary (line 2) and it extracts all the genes from the set of gene sequential patterns to be aggregated (line 3), *i*.e. it computes the union of all the gene sequential patterns. Following, the annotated set of the union is calculated (line 4), this makes us possible to know if a concrete gene is stored in all gene sequential patterns or not. Then, the average relative positions of genes are computed (line 5), this allows us to know where the genes have to be placed in the summary. Following, we build a summary considering the genes union. If a certain gene is found in all sequences (lines 7-8), it is added to the summary. Note that, if a gene is placed in all the sequences in the same position, then it is placed exactly in this place (function $add(p, g)$ allows us to add gene g in the position p). If a gene is not stored in all the sequences (line 10), it is replaced by a wild-card. Finally, all wild-cards are resumé. Example 1 illustrates this algorithm.

Algorithm 1: Summarization Method

```
   Data: GS: Set of Gene Sequential Patterns
   Result: Sum: Gene Sequential Pattern Summary
 1 begin
 2     Sum = {∅}
 3     Union = GenesExtraction(GS)
 4     Counter = GenesCount(Union, GS)
 5     Pos = GenesPosition(Union, GS)
 6     for i← 1 to Union.Size do
 7         if Counter.get(i) = GS.Size then
 8             Sum.add(Pos.get(i),Union.get(i))
 9         else
10             Sum.add(Pos.get(i),wild-card)
11     Sum=resuméWild-cards(Sum)
12     return Sum
13 end
```

Example 1 *Let $S =< G_A, G_B, G_C, G_D, G_F >$ and $S' =< G_D, G_A, G_C, G_E >$ be two different gene sequential patterns. The first step of our summarization procedure is to compute the union of these two sequences, $S \cup S' =< G_A, G_B, G_C, G_D, G_E, G_F >$, following we compute the annotated set of $S \cup S'$ as $\overline{S \cup S'} =< 2, 1, 2, 2, 1, 1 >$ and the average relative positions as $\widehat{S \cup S'} =< \frac{(1/5)+(2/4)}{2}, 2/5, \frac{(3/5)+(3/4)}{2}, \frac{(4/5)+(1/4)}{2}, 4/4, 5/5 >$. Now, we build the summary adding each gene of the union in the correct order using the function $add(p, g)$ and replacing the non common genes (genes with a value different from 2 in $\overline{S \cup S'}$) by a wild-card, in this example $(*)$. Doing this, we obtain*

$\mathcal{S} =< G_A, *, G_D, G_C, *, * >$. *Finally, wild-cards are summarized obtaining* $\mathcal{S} =< G_A, *, G_D, G_C, * >$.

3.2 $OWAGen$ Similarity Function

Formally, a similarity function f over two sequences S and S' has to fulfill the following conditions: (i) Symmetry: $f(S, S') = f(S', S)$, (ii) Positivity: $f(S, S') \geq 0$ for any S and S', and (iii) Reflexivity: $f(S, S) = 0$.

Some similarity measures for sequences of symbols have been defined. As explained in [4], these similarity measures are not adapted for gene sequential patterns. For instance, Hamming distance or Edit (Levenstein) distance are usually considered [2]. However, such distances are quite simple when it is necessary to deal with the relative ordering positions of the symbols in the sequence. Another drawback of traditional sequence distances is the difficulty of incorporating user (expert) knowledge.

As we are interested in comparing gene sequential patterns, those distances have been not considered since they disregard a lot of information that should be taken into account in order to obtain a more appropriate similarity value. Specifically, there are two key issues to be considered: the number of common genes and the order (position) among the common genes inside the sequences.

On one hand, the first consideration allows us to evaluate the common part of two different sequences, if this common part is large, then the two sequences have to be similar, because such sequences show that their share a large number of genes. On the other hand, the second consideration takes into account whether the genes have been placed in the same order or not. This second information is very useful because genes order has several biological properties. Note that in this latter case, if two sequences have the same genes placed in the same order, they have to be more similar than two sequences in which the genes are placed in different order.

Apart from these two issues, it is also interesting to incorporate any user (expert) preferences on the similarity function. In some scenarios, it is more important to consider as more similar sequences sharing a large number of genes even though the genes are not in the same order. On the contrary, in another scenarios, finding the largest sub-sequences of common genes is more interesting, considering that two sequences with a few genes in the same order are more similar than two sequences with a larger common set of unsorted genes.

OWA operators can be used to introduce the expert preferences into the computation of the similarity measuse. This allows us to tackle the issues explained before. Based on this, a new similarity function can be defined, OWAGene. It is calculated in two steps.

Firstly, partial distances among the genes of the two sequential patterns are calculated as follows.

Definition 3 *Let S and S' be two gene sequential patterns. Let G_i be the i th gene of the sequence S, then the gene distance of G_i with respect to S' is defined as*

$$d(G_i, S') = \begin{cases} \frac{|i-j|}{|S \cup S'|} & if\ G_i \in S'\ in\ the\ j\ th\ position \\ 1 & otherwise \end{cases} \quad (1)$$

where $|S \cup S'|$ stand for the number of different genes in sequences S and S'. Note that a gene G_i can only appear once in each sequence, then the distance $d(G_i, S')$ is very easy and fast to compute.

Example 2 *Let S and S' be the same gene sequential patterns than Example 1. Let $S \cup S' =< G_A, G_B, G_C, G_D, G_E, G_F >$ be the union of S and S'. The cardinality $|S \cup S'|$ is equal to 6. Now, Let us compute the partial distance of the gene G_A, such gene is placed in the position number 2 in S, whereas it is placed in the position number 1 in S', then, the partial distance $d(G_A, S')$ is computed as $d(G_A, S') = \frac{|2-1|}{6} = 0.167$.*

Those partial distances are aggregated using the OWA operator, which permits to apply different aggregation policies. Formally, given two gene sequential patterns S and S', we compute the gene distances $d(G_i, S')$ for all $i = 1, \ldots, |S|$ and $d(G_i, S)$ for all $i = 1, \ldots, |S'|$. Then, OWAGene is defined as follows.

Definition 4 *Let S and S' be two gene sequential patterns with length $|S|$ and $|S'|$, respectively, then the $OWAGene$ similarity function of S and S' is defined as*

$$OWAGene(Q)(S, S') = OWA_Q(d(G_1, S'), \ldots, d(G_{|S|}, S'), d(G_1, S), \ldots, d(G_{|S'|}, S))$$

where Q is a non-decreasing fuzzy quantifier, $d(G_i, S')$ stands for the gene distance of G_i with regards to S'.

Following Definition 3, all partial genes distances equal to 1 represents the case of a non common gene in two sequences. On the contrary, distances smaller than 1 represent a shift between a common gene in both sequences. In this case, the larger the genes distance, the larger the shift. OWA operators permit the user to aggregate the values giving more importance to large or small values when the appropriate quantifier is selected. This stands for considering as more important non common genes (distance values equal to 1), common genes (distance values between 0 and 1) or common genes in the same place (distance values equal 0).

Example 3 *Let us compute the* $OWAGene(S, S')$ *between the same gene sequential patterns than Example 1. Firstly, we have to select one quantifier. Here, we are interested in considering as the most important criterium that sequences share a large number of common genes. We use* $Q^{\alpha}(x) = x^{\alpha}$ *with a small* $\alpha = 0.5$*. Following we compute the partial distances between genes of sequences* S *and* S' *as* $|\widehat{S \cup S'}| =< 0.17, 1, 0, 0.5, 1, 1 >$*. After that, we have to compute the weighting vector from the selected quantifier* $Q^{0.5}$*, that is* $w =< 0.41, 0.17, 0.13, 0.10, 0.10, 0.09 >$*. Finally, we compute* $WAGene(Q^{0.5}) = 0.78$*.*

$OWAGene$ can also be applied when sequences have wild-cards. In this case, the partial distance of a non common gene is computed as the distance between such gene and the closest wild-card. This will be used in the clustering algorithms, for instance in k-means when sequences have to be associated with the closest centroid.

4 Experiments

For the experiments, we have generated a gene sequential patterns dataset from the analysis of several microarray Affymetrix DNA U133 plus 2.0. Specifically, we have 14 chips corresponding to 14 grey mouse lemurs divided into: young/old. In Table 1 we show the parameters used in our experiments as well as some clustering performance measures. First and second column show the parameters of the clustering algorithm. In particular, we show the quantifier used in the $OWAGen$ similarity function and the type of linkage (Simple, Average and Complete) respectively. The quantifier $Q^{0.5}$ gives more importance to common genes, whereas Q^2 gives more importance to common genes in the same order. Third column presents the average distance between all the leaves and the root node. Following, fourth column shows the average size of each node inside the hierarchic tree. Finally in the last column, we show the average number of wild-cards inside the nodes of the hierarchic tree.

Q	Link Type	Path distance	Node size	Wild-cards
$Q^{0.5}$	S	2.32	4.68	0.99
	A	2.22	4.84	0.96
	C	2.24	4.14	0.95
Q^2	S	2.32	4.73	1.01
	A	2.21	4.92	0.96
	C	2.24	4.16	0.95

Table 1: Average results for the hierarchical gene sequential pattern clustering.

Observing Table 1, we can deduce that using the complete linkage parameter we obtain smaller intermediate nodes and with less wild-cards than using the other configurations (simple and average linkage). This is possible because using the complete linkage configuration we are enforcing that only gene sequential patterns closer to all the elements of one cluster are added. The final size of the cluster determines in some way the number of wild-cards.

5 Conclusions

We have introduced a new similarity function and summarization algorithm for gene sequential patterns. We have implemented them into a clustering algorithm.

References

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. of the 11th ICDM*, pages 3–14, 1995.

[2] G. Dong and J. Pei. *Sequence Data Mining*. Springer, 2007.

[3] F. Hoerndli, D. David, and J. Götz. Functional genomics meets neurodegenerative disorders: Application and data integration. *P. in Neurobiology*, 76(3):169–188, 2005.

[4] H. Saneifar, S. Bringay, A. Laurent, and M. Teisseire. S2mp: Similarity measure for sequential patterns. In *AusDM*, pages 95–104, 2008.

[5] R. R. Yager. Families of OWA operators. *Fuzzy Sets and Systems*, 59:125–148, 1993.